INRet: A General Framework for Accurate Retrieval of INRs for Shapes

Yushi Guan[1], Daniel Kwan[1], Ruofan Liang[1], Selvakumar Panneer[2],
Nilesh Jain[2], Nilesh Ahuja[2], Nandita Vijaykumar[1]
[1]University of Toronto [2]Intel
{guanyushi, dkwan, ruofan, nandita}@cs.toronto.edu
{selvakumar.panneer, nilesh.jain, nilesh.ahuja}@intel.com

Abstract

Implicit neural representations (INRs) have become an important method for encoding various data types, such as 3D objects or scenes, images, and videos. They have proven to be particularly effective at representing 3D content, e.g., 3D scene reconstruction from 2D images, novel 3D content creation, as well as the representation, interpolation and completion of 3D shapes. With the widespread generation of 3D data in an INR format, there is a need to support effective organization and retrieval of INRs saved in a data store. A key aspect of retrieval and clustering of INRs in a data store is the formulation of similarity between INRs that would, for example, enable retrieval of similar INRs using a query INR. In this work, we propose ***INRet*** *(****INR Ret****rieve), a method for determining similarity between INRs that represent shapes, thus enabling accurate retrieval of similar shape INRs from an INR data store. INRet flexibly supports different INR architectures such as INRs with octree grids, triplanes, and hash grids, as well as different implicit functions including signed/unsigned distance function and occupancy field. We demonstrate that our method is more general and accurate than the existing INR retrieval method, which only supports simple MLP INRs and requires the same architecture between the query and stored INRs. Furthermore, compared to converting INRs to other representations (e.g., point clouds or multi-view images) for 3D shape retrieval, INRet achieves higher accuracy while avoiding the conversion overhead.*

1. Introduction

Implicit neural representations (INRs) have become an important approach for representing various types of data, including images, videos, audio, and 3D content [14, 26, 40, 54]. Instead of storing the data explicitly (such as RGB pixel values for images or meshes for 3D data), INRs typically encode the data implicitly by training a neural network that maps an input location to an output value. Compared to traditional representations, INRs offer several key advantages including a compact and differentiable representation, and the ability to be decoded at any resolution [8, 30]. INRs have seen many applications including neural compression [14, 54], super-resolution, and super-sampling for images and videos [7, 9]. More importantly, INRs have emerged as a promising approach for learning and representing 3D content, including learning 3D neural radiance field (NeRF) from 2D images for novel view synthesis, combining with image diffusion models for 3D model generation, as well as the representation, interpolation and completion of 3D shapes [26, 30, 32]. Given the promising advantages of INRs, they are expected to become an important format for representing and storing 3D visual data. As more and more 3D visual data are generated in this format, we need a way to store, organize, and retrieve them as required.

A key aspect of organizing and retrieving INRs in a data store involves defining a similarity measure between INRs. This enables the retrieval of any stored INR from a database or data store using a query, such as an image or another similar INR. For instance, accurate INR retrieval can facilitate finding a similar but more detailed model or alternative recommended models from a collection of AI-generated 3D models or reconstructed models of real scenes. Retrieval of similar INRs from a store can also be potentially used in content creation pipelines [15], for applications such as scene completion [38] and shape enhancement [3]. Prior research has explored methods to determine similarity and retrieve 3D models represented by traditional formats like point clouds [33–35], meshes [16, 22, 27], and voxel grids [46, 47]. These approaches typically involve encoding shapes into embeddings using deep neural networks, where the similarity between shapes is indicated by the cosine similarity of their embeddings. However, there is little research on determining similarity and enabling the retrieval of INRs in a data store.

In this work, our goal is to design a method that accurately and efficiently determines the similarity between

INRs that represent shapes in a data store. This enables accurate retrieval of similar shapes (represented as INRs) from an INR data store using a query INR, as well as the clustering and organization of INRs representing similar shapes in the data store.

There are several challenges in enabling the accurate retrieval of shapes represented as INRs. Firstly, INRs can have many different architectures. For example, INRs can be multilayer perceptrons (MLPs) with different activation functions [30, 39] or more commonly, a combination of MLPs and different types of spatial feature grids, such as octrees [42], triplanes [4], and hash grids [29]. The only prior works that enable determining similarity between INRs only support an MLP-based architecture [13, 36], which is less commonly used today due to their limitation in representation capacity and training speed. This raises a challenge in how to flexibly support MLP-only INRs, feature grid-based INRs, and possibly any future architectures the query/stored INRs may have. Secondly, INRs can represent similar shapes using different implicit functions. For example, to represent 3D shapes, signed distance function (SDF), unsigned distance function (UDF), and occupancy field (Occ) are common implicit functions that INRs can encode [11, 25, 30]. Different implicit functions are used because of their different advantages and heterogeneity of the source 3D data. First, these functions are capable of capturing different information of the same shape. For example, as we will demonstrate in Fig. 3, a car can be represented as a multi-layered car using UDF, capturing more intricate details *inside* the car. The same car can also be represented with an SDF, such a representation can facilitate the creation of watertight surfaces that clearly delineate the car's interior from its exterior. This characteristic of SDFs is particularly advantageous for applications requiring clear boundary definitions, such as voxelization and marching cubes [18, 23]. Second, the heterogeneity of source data often dictates the choice of implicit function. For example, UDF INRs are most commonly used to encode point clouds [11, 55]. Different shape generation methods can also produce different representations; Pix2Vox generates voxel grids [51], while other efforts like SDFusion and AutoSDF opt for SDF to represent shapes [10, 28]. Therefore, it is essential to develop a general method that is effective across a broad spectrum of INR architectures and implicit functions, a solution that has yet to be achieved in current literature.

In this work, we investigate different approaches to enable 3D shape retrieval and similarity estimation for INRs while addressing these challenges. We propose INRet, a framework that enables identifying similar INRs and thus accurate retrieval of similar INRs given a query INR. INRet flexibly supports INRs that use any architecture as well as any implicit function. We also compare INRet with retrieval using existing approaches for traditional representations (for example, point clouds and multi-view images) by simply converting the shape INRs into these representations for retrieval.

With INRet, we determine the similarity between INRs by first generating embeddings for each INR. The similarity between these INRs is then estimated by calculating the cosine similarity of their embeddings. We now describe how we generate embeddings that are general across different INR architectures and implicit functions, allowing for comparison in a common embedding space.

First, we design an INR Embedding Encoder (hereafter referred to as *Emb. Encoders*) that generates an embedding from the weights of the INR MLP and the learned parameters of the INR's feature grid. The key idea behind *Emb. Encoder* is to encode the MLP and feature grid of the INR using an MLP encoder and a Conv3D encoder respectively. This encoder supports both MLP-only INRs and INRs with feature grids. The Conv3D encoder can take in learned feature vectors from any spatial grid as input. In this work, we demonstrate this generality with octree grids (NGLOD), triplanes (EG3D), and multi-resolution hash grids (iNGP) [4, 29, 42]. The embeddings created from these encoders are then concatenated to create the INR embedding for shape retrieval. To train these encoders, we also simultaneously train a *Shape Decoder* that receives the INR embedding as input and outputs the INR's implicit function. The decoder is trained to approximate the implicit function values of the INRs used for training.

Second, to support retrieval of different implicit functions, we use separate *Emb. Encoder* to encode INRs with different implicit functions. They are however trained to generate embeddings that are consistent across different implicit functions. To do this, during the encoder training process, we generate INRs and the corresponding INR embeddings, representing UDF, SDF, and Occ for *each* training 3D shape. We then introduce two key regularization techniques to unify the embedding space. The first technique involves applying an explicit L2 Loss to reduce the discrepancies between embeddings of INRs that (despite their different implicit functions) represent the same shape. The second regularization is to use a *Unified Shape Decoder* that outputs a single type of implicit function value (such as UDF) for all three implicit functions. The *Shape Decoder* is only used to assist the training of *Emb. Encoders*, by using a *Unified Shape Decoder*, we improve the generality of the INR embedding created across different implicit functions. Our findings indicate that the *Unified Shape Decoder* approach greatly contributes to unifying the latent space. We show that applying both regularizations is essential for ensuring the high retrieval accuracy of INRs across different implicit functions.

We demonstrate the effectiveness of our solution on the ShapeNet10 and Pix3D datasets. Compared to existing

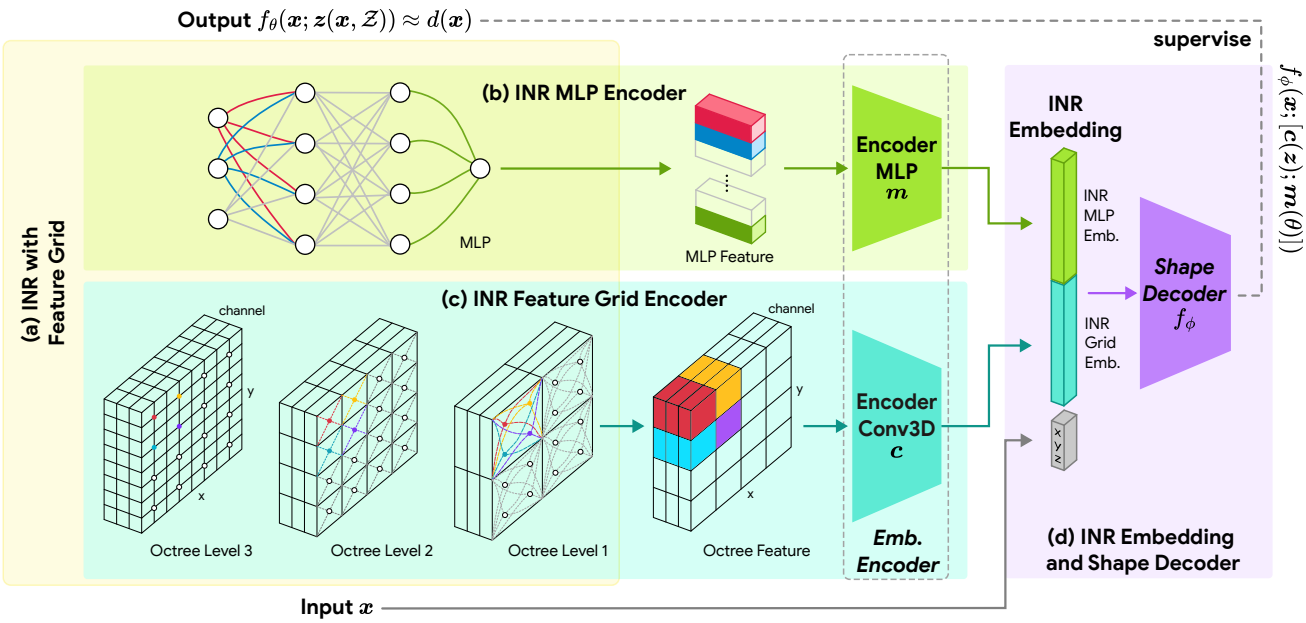


Figure 1. **Encoders trained to generate embeddings for grid-based INRs:** INRs with hash-table/octree/triplane based feature grids are used to generate embeddings for similarity calculations using encoders ($\boldsymbol{m}$ and $\boldsymbol{c}$). The encoders take MLP weights and feature grid parameters as inputs to generate the INR Embedding. During training, the encoders ($\boldsymbol{m}$, $\boldsymbol{c}$) and the decoder (f_ϕ) are jointly trained: the encoders to produce the INR Embedding and the decoder to reconstruct the original shape, ensuring the embeddings carry information about the shape. During inference, only the encoders are used to generate the INR Embedding for similarity calculations and retrieval.

methods that perform retrieval of INRs directly, our method enables retrieval of INRs with feature grids, which cannot be done with existing solutions. Our method achieves 10.1% higher retrieval accuracy on average than existing methods that can only retrieve shapes represented by MLP-only INRs [13]. We show that our regularization techniques enable retrieval of INRs across different implicit functions, achieving accuracy close to retrieval of INRs with the same implicit functions. Compared with retrieval methods applied to other representations such as point cloud and multi-view images converted from INRs, INRet achieves 12.1% higher accuracy. Except in cases where INR architecture conversion is required, INRet has lower retrieval latency as it avoids the computation overhead associated with converting to other representations.

The contributions of this work are summarized as follows:

- We pose the challenge of evaluating similarity between INRs for the retrieval and organization of shape INRs in a data store. This involves assessing various techniques, including conversion to traditional formats and direct embedding creation from INRs.
- We propose a method to create embeddings from INRs, with or without feature grids, to represent shapes for retrieval and similarity evaluation purposes.
- We propose regularization techniques to produce embeddings in a unified latent space that facilitates comparison and retrieval across INRs with different implicit functions.
- We achieve higher retrieval accuracy on both the ShapeNet10 and Pix3D datasets compared to existing INR retrieval methods and methods involving conversion to traditional representations.

2. Background & Related Works

2.1. Implicit Neural Representation for Shapes

Traditionally, 3D shapes have been represented with explicit representations including meshes, point clouds, and 3D voxels. Implicit Neural Representation (INR) has emerged as a novel paradigm for encapsulating shapes, employing neural networks to encode functions that implicitly represent a shape's surface. Seminal works like DeepSDF and Occupancy Networks demonstrate the feasibility of employing neural networks to encode signed distance function (SDF) or occupancy of 3D shapes [25, 30]. Recent advancements extended this approach to encode unsigned distance function (UDF), showcasing higher representation quality for thinner surfaces [1, 2, 11, 55].

INRs with Multi-layer Perceptron. Earlier works in INRs for shapes use simple multi-layer perceptrons (MLPs) with ReLU activations to represent the implicit functions [1, 2, 8, 11, 25, 30, 55]. SIREN proposed to use sinusoidal activation functions in MLPs to more efficiently encode higher frequency details [39]. Since the implicit function is encoded in a single MLP, the MLP is usually relatively

big and expensive to evaluate. The training of these MLPs to accurately represent the shapes is also time-consuming.

INRs with Spatial Feature Grids. While overfitting a large MLP to a shape can be difficult and computationally expensive, recent INRs for shapes use a combination of smaller MLPs and feature grids with learnable parameters. Peng et al. introduced Convolutional Occupancy Networks, which combine a trainable 3D dense feature grid and an MLP [31]. Recent works have extended this notion and applied multi-level feature grids to encode and combine information at varying levels of detail. These multi-level spatial grids can be represented as sparse octrees as seen in NGLOD, VQAD, NeuralVDB, and ROAD [20, 42, 43, 53]. EG3D and iNGP introduced the idea of using triplanes multi-level hash grids to store these features at a fixed memory budget [4, 29]. Sparse octrees, triplanes, and hash grids have seen wide applications in implicit neural representations for shapes or for radiance fields [6, 45, 52]. Compared to INRs with only MLP, they significantly improve representation quality, as well as training and rendering speed. Our method considers INRs with or without the spatial grid for retrieval. We do so by optionally encoding the spatial grid for the INR embedding creation.

2.2. Shape Retrieval

INR Retrieval. Numerous techniques have been developed to encode 3D shapes using INRs. The seminal work DeepSDF employs a shared Multi-Layer Perceptron (MLP) with varying latent codes to represent distinct shape instances [30]. These latent codes can be used for shape retrieval, as akin shapes tend to exhibit similar codes. Nonetheless, the adoption of the shared MLP concept in subsequent research has been limited due to its compromised representation quality when contrasted with employing a dedicated MLP for each shape [12]. The retrieval of dedicated MLP INRs has been studied in [13]. However, MLP-only INRs still fall behind INRs with feature grids in terms of representation quality and training speed. The method allowing the retrieval of INRs with spatial feature grids and/or across INRs with different implicit functions has yet to be developed.

Retrieval by Converting to Traditional Representations. Shape retrieval for traditional 3D representations has many established works with techniques proposed for voxels, meshes, point clouds, and multi-view images [22, 33, 46, 48]. However, these methods do not directly apply to the retrieval of INRs. A viable approach to retrieve INRs for shapes is to first transform these representations into one of the aforementioned traditional representations, and then apply established retrieval methods. For comparison with our method, We select two representations: point clouds and multi-view images, as they achieve higher accuracy in retrieval compared to other traditional representations [22, 46]. Besides higher accuracy, point-based and multi-view image-based methods also avoid the computational overhead of the voxel-based methods and the requirement for watertight surfaces for the mesh methods [27, 47].

We use the state-of-the-art methods PointNeXt and View-GCN as point-based and multi-view images-based baselines for comparison [35, 48].

3. Methods

3.1. Preliminary - INR for Shapes

In this section, we introduce the different INR implicit functions and architectures we consider in this work. Consider a general distance or occupancy function $d(\cdot)$, defined for input coordinates $\boldsymbol{x} \in \mathbb{R}^3$ on the input domain of $\Omega = \{\|\boldsymbol{x}\|_\infty \leq 1 | \boldsymbol{x} \in \mathbb{R}^3\}$. The goal of INR for shape is to approximate $d(\cdot)$ by a function f_θ parameterized by a neural network.

$$f_\theta(\boldsymbol{x}) \approx d(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \Omega. \tag{1}$$

Popular choices for the implicit function include signed distance function (SDF, $d_s(\cdot) \in \mathbb{R}$), unsigned distance function (UDF, $d_u(\cdot) \in \mathbb{R}^+$), and occupancy fields (Occ, $d_o(\cdot) \in \{-1, 1\}$) [11, 25, 30]. INRs are trained to minimize the difference between $f_\theta(\boldsymbol{x})$ and $d(\boldsymbol{x})$. Earlier works parameterize the function with a multi-layer perceptron (MLP). More recent works combine a feature grid with a smaller MLP, where the MLP takes the feature $\boldsymbol{z}$ sampled from the feature grid $\mathcal{Z}$ as input.

$$f_\theta(\boldsymbol{x}; \boldsymbol{z}(\boldsymbol{x}, \mathcal{Z})) \approx d(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \Omega. \tag{2}$$

The feature grid $\mathcal{Z}$ has various forms including sparse voxel octree, triplane, and hash grids, for which we all consider in this work [4, 29, 42]. All of these grids can be multi-level, encoding features at different spatial resolutions. For a multi-level feature grid, at each level $l \in \{1, ..., L\}$, the feature vector $\boldsymbol{\psi}(\boldsymbol{x}; l, \mathcal{Z})$ is interpolated (i.e., trilinearly) from local features. The final feature vector $\boldsymbol{z}$ from the grid is a summation (octree, triplane) or concatenation (hash grid) of features from all levels. The feature vector is then optionally concatenated with the input coordinate $\boldsymbol{x}$ and fed to a shallow MLP to calculate the distance or occupancy value.

3.2. Embedding Creation for INR with Feature Grids

We determine the similarity between 3D shapes represented as INRs by converting each INR into an embedding, and the similarity between shapes is determined by the cosine similarity between these embeddings. We demonstrate our process for creating embeddings from INRs with feature grids in Fig. 1. Given a trained INR with an MLP component parametrized by θ and a feature grid $\mathcal{Z}$, we use an

MLP Encoder $\boldsymbol{m}$ and Conv3D Encoder $\boldsymbol{c}$ to encode the features from the MLP and feature grid components of the INR respectively. Collectively, the MLP encoder $\boldsymbol{m}$ and Conv3D Encoder $\boldsymbol{c}$ constitutes the *Emb. Encoder*. If the INR only contains an MLP component, we can simply omit the Conv3D Encoder $\boldsymbol{c}$. For the INR MLP component, the flattened weights of INR's MLP become input vectors to the MLP encoder. The structure of the encoder MLP is described in App. 7.2.

For the INR feature grid, we sample $(2N)^3$ feature vectors at a fixed resolution from the feature grid, i.e. $S = \{[x_1 x_2 x_3]^T | x_i = \pm(\frac{1}{2N} + \frac{n}{N}), \forall n \in \{1, 2, \ldots, N-1\}\}$. The sampled features are used as inputs to the Conv3D encoder, a 3D convolutional network that fuses discrete spatial features with gradually increasing perception fields (see Appendix 7.2 for more details). We use an octree (visualized in 2D) as an example in Figure 1**(c)**. Depending on the sampling resolution and resolution of the octree level, the feature is either collected directly from the corners of the voxels (Octree Level 3 in the example), or interpolated using features stored at the corners of the voxel containing the sampling location (Octree Level 2 & 1). The features collected from each level are summed together, simply adding zero if a voxel is missing (due to sparsity in the octree). The collected features are fed to a Conv3D Encoder to create the INR Grid Embedding. A similar summation process can be done by traversing through the triplane grid levels. For a multi-resolution hash grid-based INR, we retrieve the features directly at the sampled locations using the original hash function and hash table. The MLP embedding and grid embedding are then concatenated to create our INR embedding.

Training *Emb. Encoders*. During the encoder training process, we feed the concatenation of the INR embedding and the input coordinate $\boldsymbol{x}$ to the *Shape Decoder* f_ϕ. The decoder is supervised to generate the original implicit function that represents the shape. Thus, the encoders are trained to generate embeddings that can be used to regenerate the original shape using the decoder. The following equation describes the process, where the decoder f_ϕ approximates the implicit function value of the original shape:

$$f_\phi(\boldsymbol{x}; [\boldsymbol{c}(\boldsymbol{z}); \boldsymbol{m}(\theta)]) \approx d_{i \in s,u,o}(x) [\approx f_\theta(\boldsymbol{x}; \boldsymbol{z}(\boldsymbol{x}, \mathcal{Z}))]. \tag{3}$$

Note that since the INR parametrized by $\boldsymbol{\theta}, \boldsymbol{z}$ also encodes the implicit function, the INR Embedding $[\boldsymbol{c}(\boldsymbol{z}); \boldsymbol{m}(\theta)]$ is trained to contain information of the original shape.

Supporting other INR architectures. INRet assumes a separate encoder for each type of INR architecture that is supported. Our proposed encoders already support the commonly used octree-based, triplane, and hash grid INR architectures. A similar feature grid sampling approach can be used to also train an encoder for any new grid-based architecture. Alternatively, other architectures can still be used with the above two encoders by using a distillation technique that converts a new INR architecture into one of the representations that we support. We describe how this can be done in App. 7.3.

3.3. Unified Latent Space for INRs with different Implicit Functions

Besides different architectures, INRs can encode different implicit functions for the same underlying shape. To support multiple implicit functions of the same shape, we train separate encoders for each implicit function. To ensure that the generated embeddings map to the unified latent space, we apply two regularization techniques during the encoder training process.

The first regularization applied is explicit L2 loss to minimize the difference between embeddings created from INRs for different implicit functions of the same shape. The second regularization is to use a *Unified Shape Decoder* that outputs a single type of implicit function value (such as UDF) for all three implicit functions. We show in App. 8.4 that the specific choice of *Unified Shape Decoder* implicit function (UDF *vs.* SDF *vs.* Occ) has minimal impact on the retrieval accuracy. The key is that both regularizations are applied.

The overall loss function for this process is

$$\mathcal{L} = \sum_{i \in \{s,u,o\}} |f_\phi(\boldsymbol{x}; \boldsymbol{e}_i) - d_u(\boldsymbol{x})| + \lambda \sum_{i,j \in \{s,u,o\}} \|\boldsymbol{e}_i - \boldsymbol{e}_j\|^2 \tag{4}$$

$\boldsymbol{e}_i = [\boldsymbol{c}_i(\boldsymbol{z}_i); \boldsymbol{m}_i(\theta_i)]$ is the INR embedding for the implicit function i (unsigned/signed distance or occupancy). The first part of the loss is the difference between the *Unified Shape Decoder's* output with the groundtruth unsigned distance d_u, and the second part is the L2 loss between the INR embeddings. λ is a hyperparameter balancing the contribution of the two parts, we found a λ of 1 works well in practice. During the encoder training process, we create INRs for all implicit functions of each training shape to train the encoders to generate embeddings that share the unified latent space.

4. Retrieval by Converting to Explicit Representations

An alternative approach to evaluate similarity and enable retrieval of similar INRs is to first convert to an explicit representation, such as point clouds or multi-view images. This approach would enable the use of prior research to evaluate similarity between shapes represented in these traditional formats. In this work, we also evaluate the effectiveness of this approach in comparison to directly using INR embeddings. Conversion to point clouds and multi-view images from SDF INRs can be done through spherical tracing [17]. For point cloud sampling, we start spheri-

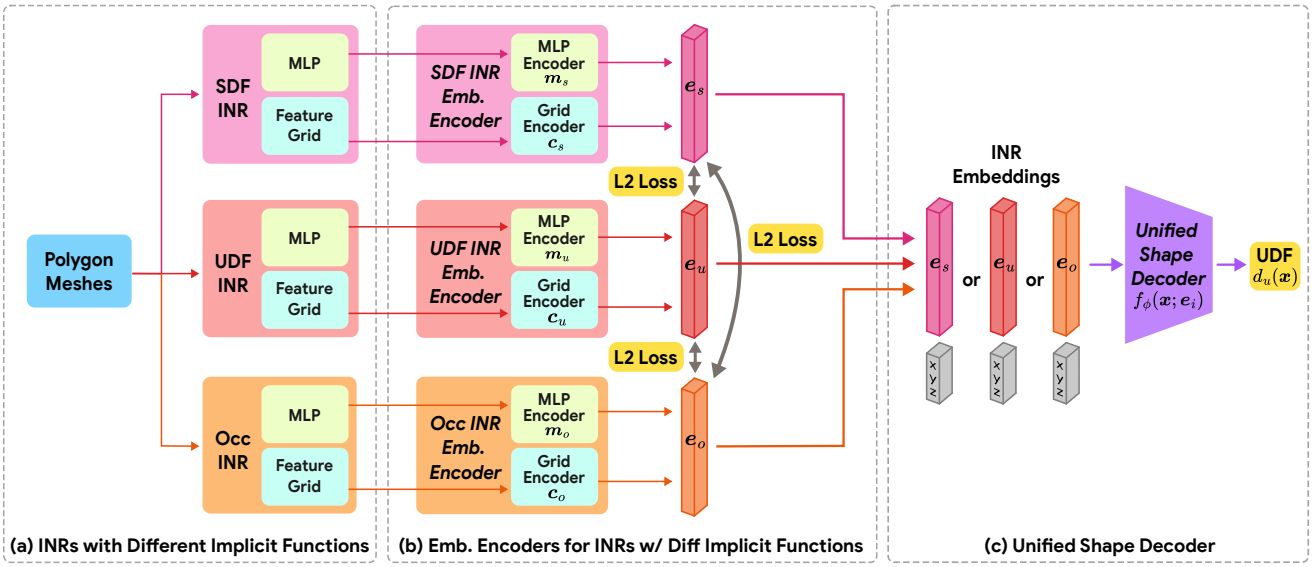


Figure 2. **INR Embed. Creation for INRs with Different Implicit Functions.** (a) For each shape, we train INRs with different implicit functions. (b) We train different encoders for INRs with different implicit functions. The differences between embeddings created by the encoders are minimized by L2 loss. (c) We feed the embeddings into a *Unified Shape Decoder* to recreate the UDF of the original shape.

cal tracing from randomly selected locations and directions until enough points on the surface of the object are collected [42]. The multi-view images are also collected via spherical tracing starting from camera centers at fixed positions. For UDF INRs, we use the damped spherical tracing presented in prior work [11] that avoids overshooting. For the occupancy values, spherical tracing is not possible so we follow the method presented in Occupancy Networks [25]. Using occupancy values sampled at fixed resolutions from the trained INR, we combine isosurface extraction and the marching cubes algorithm to create the surface mesh of the object [23]. We then perform point cloud sampling and multi-view image rendering from the constructed mesh. To generate embeddings for similarity evaluations from these formats, we use PointNeXt [35] for extracted point clouds, and View-GCN [48] for multi-view images (details are in App. 7.4).

5. Evaluation

5.1. Experimental Setup

Datasets. We use ShapeNet and Pix3D to demonstrate the generality and robustness of our solution [5, 41]. For the ShapeNet10 dataset, each category has 50 models for training and 50 models for testing. For Pix3D, we use 70% of the shapes from each category as training data and 30% as testing data.

Metrics. We evaluate the effectiveness of our framework in identifying similar shapes in the data store by using a test INR shape to retrieve the most similar k INR shapes. We report the mean Average Precision (mAP) as the average accuracy of retrieving a shape from the same category as the query shape across all shapes in the test set. We also report precision, recall, and F1 score as defined in the ShapeNet retrieval challenge in the App. 8 [37].

Baselines. We compare against inr2vec for retrieval from INRs by directly encoding the INR weights. We also compare with PointNeXt and View-GCN by converting the trained INR to point-cloud and multi-view images, respectively.

Ablations. We only report key results in this section, and provide additional results and more detailed ablation studies in App. 8.

5.2. INR Retrieval with Feature Grids

To create a baseline shape INR data store, we train NGLOD (octree), EG3D (triplane) and iNGP (hash-grid) INRs with SDF to encode shapes from ShapeNet10 and Pix3D datasets [4, 29, 42]. The encoders are trained on our training set and used to generate embeddings for the test set. Tab. 1 presents mAP@1 and retrieval speed (in seconds), and additional metrics are available in Appendix 8.1. Our comparison includes INRet against inr2vec, which performs retrieval on MLP-only INRs of the same implicit function. Additionally, we compare with PointNeXt and View-GCN by converting the trained iNGP INR to point clouds and multi-view images for retrieval.

As seen in Tab. 1, INRet achieves the highest accuracy: on average 12.0%, 15.4%, and 12.6% higher accuracy than inr2vec, PointNeXt and View-GCN methods respectively

Method	Ours			inr2vec	PointNeXt	View-GCN
Input Type	NGLOD	EG3D	iNGP	MLP INR	Point Cloud	Multi-View
mAP @ 1	82.6/74.3	82.8/74.1	**84.2/78.0**	73.4/71.4	71.2/69.3	73.6/70.5
Ret. Speed(s)	0.034	0.031	0.14	0.062	0.98	3.05

Table 1. Shape Retrieval Accuracy and Speed on SDF INRs (ShapeNet10/Pix3D)

Method	Ours			PointNeXt	View-GCN
Input Type	NGLOD	EG3D	iNGP	Point Cloud	Multi-View Images
mAP @ 1	76.2/71.3	76.4/70.4	**79.2/75.5**	70.2/67.1	71.4/68.2
Ret. Speed(s)	30.2	34.1	29.6	1.26	4.57

Table 2. Shape Retrieval Accuracy with MLP-only INR as Query (ShapeNet10/Pix3D)

(for iNGP INRs). For INRet, retrieving from iNGP INRs achieved slightly higher performance than retrieving from NGLOD and EG3D INRs. In terms of retrieval speed, INRet on NGLOD and EG3D are the fastest, followed by inr2vec, which is slightly slower due to the large number of weights in the INR MLP. Compared to NGLOD and EG3D, from which the embedding can be directly summed from the feature grid, embedding sampling from the iNGP hash-grid is slower due to the higher overhead of the hash operations during sampling. Converting to point clouds or multi-view images for retrieval with PointNeXt or View-GCN is 1-2 orders of magnitude slower than directly encoding the INR weights.

In summary, INRet enables high-accuracy retrieval of similar shape INRs. Converting to images or point clouds leads to lower accuracy due to information loss during the conversion process and incurs the latency overhead for format conversion.

5.3. INR Retrieval with Different Architectures

In this section, we evaluate INRet's effectiveness in retrieving shapes across different INR architectures. Given an MLP-only INR as the query, we want to retrieve from a data store of INRs with different architectures from the query INR. We consider an MLP INR similar to that used by inr2vec as input. We apply the INR distillation technique discussed in Sec. 3.2 (Supporting other INR architectures) to convert the MLPs into INRs with feature grids to retrieve NGLOD, EG3D or iNGP INRs.

As depicted in Tab. 2, following INR distillation, INRet achieves an average accuracy of 73.8%, 73.4% and 77.4% respectively for NGLOD, EG3D and iNGP *Emb. Encoders* across the two datasets, surpassing the average accuracy of 72.4% achieved by inr2vec. Our method also performs better than converting to point cloud or multi-view images. This highlights the robustness of our approach in adapting to different architectures not directly supported by the encoder. Despite performing a distillation, the converted INRs with a feature grid can be used to generate better embeddings for retrieval when compared with generating the embeddings directly from MLP-only INR. While format conversion introduces some overhead (approximately 30 seconds), a potential speed-accuracy tradeoff could be explored by converting to point clouds/images when INRet lacks a pre-trained encoder for a new architecture.

		Retrieval INR		
		UDF	SDF	Occ
Query	UDF	80.2/80.8/83.0 68.8/72.0/70.8	81.4/79.4/79.0 10.4/61.8/72.6	78.8/79.2/80.4 8.8/58.2/68.4
	SDF	82.2/81.2/81.8 11.4/62.2/70.2	83.4/82.4/84.6 70.2/67.2/69.4	79.2/79.6/82.4 10.4/56.2/68.8
	Occ	76.0/79.8/81.0 9.2/55.4/62.6	77.0/79.4/82.6 10.4/56.2/61.8	76.8/80.0/83.0 69.4/51.2/66.4
Average		79.4/80.2/**82.0** 29.9/60.0/67.9		
Legend		*Ours*: NGLOD/ EG3D / **iNGP** *Baselines*: inr2vec/PointNeXt/View-GCN		

Table 3. Shape Retrieval Accuracy on Different Implicit Functions INRs for ShapeNet10

		Retrieval INR		
		UDF	SDF	Occ
Query	UDF	83.4/83.4/83.0	9.4/52.4/79.0	10.8/51.8/80.4
	SDF	10.8/57.8/81.8	82.4/81.4/84.6	9.6/53.2/82.4
	Occ	11.4/65.4/81.0	10.2/53.2/82.6	81.6/82.4/83.0
Average		34.0/64.6/**82.0**		

Table 4. Shape Retrieval Accuracy on iNGP INRs for ShapeNet10 (No Reg. / L2 Reg. / **L2 Reg.** & ***Unified Shape Decoder***)

5.4. INR Retrieval with Different Implicit Functions

In this section, we evaluate the effectiveness of our method in performing INR retrieval across INRs with different implicit functions (i.e., UDF, SDF and Occ). We compare against inr2vec and point-based and image-based methods.

As seen in Tab. 3, using our method to retrieve iNGP INRs with different implicit functions achieves the highest 82.0% accuracy, which is higher than the accuracy achieved

with inr2vec, PointNeXt, and View-GCN. In particular, inr2vec achieves very low accuracy (around 10%) for retrieving INRs with different implicit functions. As seen in Tab. 4, using INRet to retrieve iNGP with different implicit functions also achieves very low accuracy (around 10%) if no regularization is used. The average accuracy for retrieval improves significantly if the L2 regularization (64.6% accuracy) and both regularizations (82.0% accuracy) are applied.

In this section, we used the original meshes to sample SDF, UDF and Occ values to train the INRs with different implicit functions. However, for the UDF INRs, one can also train the INRs given an input point cloud, we demonstrate in Appendix 10 that the retrieval accuracy does not change significantly compared to when the UDF INRs are trained using the meshes. This demonstrates that by enabling retrieval from INRs with different implicit functions with INRet, we ultimately enable retrieval of INRs trained with different 3D input modalities.

5.5. Retrieval Visualization

We visualize the retrieved shapes in Fig. 3. In particular, we demonstrate our solution enables the retrieval of INRs with different implicit functions, which is not possible with other baseline solutions.

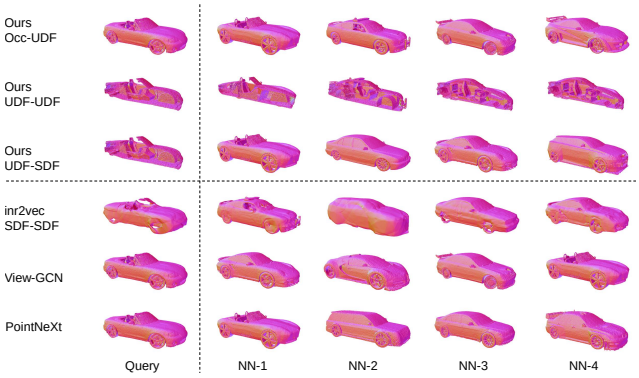


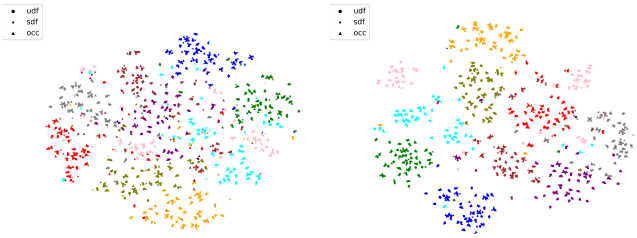
Figure 3. Retrieval Qualitative Comparison.

Fig. 3 shows the query and retrieved results for the car class in ShapeNet. As we can see from the figure, given a convertible as the query, our method consistently retrieves the other convertibles from the dataset as the top candidates while most other methods fail to do so. Fig. 3 also demonstrates INRet's ability to retrieve watertight surfaces (represented with SDF INR) from surfaces with multiple inner layers (represented with UDF INR). For the "Ours UDF-UDF" and "Ours UDF-SDF" rows, we show renderings of the same query car, but with the middle cut open when the shape is represented using a UDF INR. We can see that the UDF INR can capture the details inside the car. When used as the query, these UDF INRs can retrieve cars represented with different implicit functions correctly. In addition, compared with the renderings demonstrated in the row "inr2vec SDF-SDF" which used an MLP-only INR to represent the underlying shape. Our method uses iNGP to represent the shape thus capturing more details. We provide additional quantitative results on the reconstruction quality in App. 9.1.

While different implicit functions have very different values, we show in App. 7.5 that for the same underlying shape, their values are highly correlated.

5.6. Embedding Space t-SNE Visualization

In Fig. 4, we provide the t-SNE plot of the INR embeddings created by the *Emb. Encoders* trained with and without the *Unified Shape Decoder*. When trained using different *Shape Decoders*, the embeddings for shapes belonging to the same category are much more spread out, this is likely due to the different decoders requiring the INR embedding for the same shape to be used for different purposes (decoding UDF, SDF, or Occ). This misses the regularization from the unified decoder that further minimizes the difference between embeddings of the same shape represented with INRs with different implicit functions.



(a) INR Emb. tSNE with different *Shape Decoders*

(b) INR Emb. tSNE with *Unified Shape Decoder*

Figure 4. INR Embedding tSNE Plot

6. Conclusion

In this work, we presented a new framework for determining similarity between INRs that can be used for accurate retrieval of INRs from a data store. We proposed a new encoding method for INRs with feature grids including the octree and hash table based grids. By using L2 loss and a common decoder as regularizations, INRet also enables the retrieval of INRs across different implicit functions. On ShapeNet10 and Pix3D datasets, INRet demonstrates more than 10% improvement in retrieval accuracy compared to prior work on INR retrieval and retrieval by conversion to point cloud and multi-view images. Compared to point cloud and multi-view image retrieval methods, INRet is also faster by avoiding the conversion overhead when retrieving INRs with same or different implicit functions.

References

[1] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020. 3

[2] Matan Atzmon and Yaron Lipman. Sal++: Sign agnostic learning with derivatives. *ArXiv*, abs/2006.05400, 2020. 3

[3] Zhen Cao, Wenxiao Zhang, Xin Wen, Zhen Dong, Yu-Shen Liu, Xiongwu Xiao, and Bisheng Yang. Kt-net: knowledge transfer for unpaired 3d shape completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 286–294, 2023. 1

[4] Eric Chan, Connor Z. Lin, Matthew Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J. Guibas, Jonathan Tremblay, S. Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3d generative adversarial networks. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16102–16112, 2021. 2, 4, 6

[5] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, L. Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *ArXiv*, abs/1512.03012, 2015. 6

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *ArXiv*, abs/2203.09517, 2022. 4

[7] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8624–8634, 2020. 1

[8] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019. 1, 3

[9] Zeyuan Chen, Yinbo Chen, Jingwen Liu, Xingqian Xu, Vidit Goel, Zhangyang Wang, Humphrey Shi, and Xiaolong Wang. Videoinr: Learning video implicit neural representation for continuous space-time super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2047–2057, 2022. 1

[10] Yen-Chi Cheng, Hsin-Ying Lee, S. Tulyakov, Alexander G. Schwing, and Liangyan Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4456–4465, 2022. 2

[11] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. *ArXiv*, abs/2010.13938, 2020. 2, 3, 4, 6, 1, 7

[12] T. Davies, Derek Nowrouzezahrai, and Alec Jacobson. Overfit neural networks as a compact shape representation. *ArXiv*, abs/2009.09808, 2020. 4

[13] Luca De Luigi, Adriano Cardace, Riccardo Spezialetti, Pierluigi Zama Ramirez, Samuele Salti, and Luigi di Stefano. Deep learning on implicit neural representations of shapes. In *International Conference on Learning Representations*, 2023. 2, 3, 4, 1, 9

[14] Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compression with implicit neural representations. *arXiv preprint arXiv:2103.03123*, 2021. 1

[15] Jiaming Gu, Minchao Jiang, Hongsheng Li, Xiaoyuan Lu, Guangming Zhu, Syed Afaq Ali Shah, Liang Zhang, and Mohammed Bennamoun. Ue4-nerf: Neural radiance field for real-time rendering of large-scale scene. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[16] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38:1 – 12, 2019. 1

[17] John C. Hart. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12:527–545, 1996. 5

[18] Jian Huang, Roni Yagel, Vassily Filippov, and Yair Kurzion. An accurate method for voxelizing polygon meshes. *IEEE Symposium on Volume Visualization (Cat. No.989EX300)*, pages 119–126, 1998. 2

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015. 1

[20] Doyub Kim, Minjae Lee, and Ken Museth. Neuralvdb: High-resolution sparse volume representation using hierarchical neural networks. *ArXiv*, abs/2208.04448, 2022. 4

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 1, 6

[22] Alon Lahav and Ayellet Tal. Meshwalker: Deep mesh understanding by random walks. *ACM Trans. Graph.*, 39:263:1–263:13, 2020. 1, 4

[23] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987. 2, 6

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017. 1

[25] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 2, 3, 4, 6

[26] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf. *Communications of the ACM*, 65:99 – 106, 2020. 1

[27] Thomas W. Mitchel, Vladimir G. Kim, and Michael M. Kazhdan. Field convolutions for surface cnns. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9981–9991, 2021. 1, 4

[28] Paritosh Mittal, Y. Cheng, Maneesh Singh, and Shubham Tulsiani. Autosdf: Shape priors for 3d completion, reconstruction and generation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 306–315, 2022. 2

[29] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 2, 4, 6

[30] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 1, 2, 3, 4

[31] Songyou Peng, Michael Niemeyer, Lars M. Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. *ArXiv*, abs/2003.04618, 2020. 4

[32] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *ArXiv*, abs/2209.14988, 2022. 1

[33] C. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2016. 1, 4

[34] C. Qi, L. Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017.

[35] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *ArXiv*, abs/2206.04670, 2022. 1, 4, 6, 2

[36] Pierluigi Zama Ramirez, Luca De Luigi, Daniele Sirocchi, Adriano Cardace, Riccardo Spezialetti, Francesco Ballerini, Samuele Salti, and Luigi Di Stefano. Deep learning on 3d neural fields, 2023. 2

[37] Manolis Savva, Fisher Yu, Hao Su, M Aono, B Chen, D Cohen-Or, W Deng, Hang Su, Song Bai, Xiang Bai, et al. Shrec16 track: largescale 3d shape retrieval from shapenet core55. In *Proceedings of the eurographics workshop on 3D object retrieval*, 2016. 6, 4, 7

[38] Yawar Siddiqui, Justus Thies, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Retrievalfuse: Neural 3d scene reconstruction with a database. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12568–12577, 2021. 1

[39] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020. 2, 3, 1

[40] Kun Su, Mingfei Chen, and Eli Shlizerman. Inras: Implicit neural representation for audio scenes. In *Neural Information Processing Systems*, 2022. 1

[41] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B. Tenenbaum, and William T. Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2974–2983, 2018. 6

[42] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11358–11367, 2021. 2, 4, 6, 1, 8, 9

[43] Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. Variable bitrate neural fields. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022. 4

[44] Towaki Takikawa, Or Perel, Clement Fuji Tsang, Charles Loop, Joey Litalien, Jonathan Tremblay, Sanja Fidler, and Maria Shugrina. Kaolin wisp: A pytorch library and engine for neural fields research. `https://github.com/NVIDIAGameWorks/kaolin-wisp`, 2022. 1

[45] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 4

[46] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn. *ACM Transactions on Graphics (TOG)*, 36:1 – 11, 2017. 1, 4

[47] Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. Adaptive o-cnn: A patch-based deep representation of 3d shapes. *arXiv: Computer Vision and Pattern Recognition*, 2018. 1, 4

[48] Xin Wei, Ruixuan Yu, and Jian Sun. View-gcn: View-based graph convolutional network for 3d shape analysis. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1847–1856, 2020. 4, 6, 2

[49] Cameron R. Wolfe and Keld T. Lundgaard. E-stitchup: Data augmentation for pre-trained embeddings. *arXiv: Learning*, 2019. 9

[50] Yuxin Wu and Kaiming He. Group normalization. *International Journal of Computer Vision*, 128:742 – 755, 2018. 1

[51] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, Shengping Zhang, and Xiaojun Tong. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2690–2698, 2019. 2

[52] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, pages 641–676. Wiley Online Library, 2022. 4

[53] Sergey Zakharov, Rares Ambrus, Katherine Liu, and Adrien Gaidon. Road: Learning an implicit recursive octree auto-decoder to efficiently encode 3d shapes. *ArXiv*, abs/2212.06193, 2022. 4

[54] Yunfan Zhang, Ties van Rozendaal, Johann Brehmer, Markus Nagel, and Taco Cohen. Implicit neural video compression. *arXiv preprint arXiv:2112.11312*, 2021. 1

[55] Junsheng Zhou, Baorui Ma, Yu-Shen Liu, Yi Fang, and Zhizhong Han. Learning consistency-aware unsigned distance functions progressively from raw point clouds. *ArXiv*, abs/2210.02757, 2022. 2, 3